

Phase-Scan Console Program For Linac Tuning

Thomas L. Owens
Fermi National Accelerator Laboratory
P.O. Box 500
Batavia, Illinois 60510

April 16, 1993

Abstract

A console program has been written which can be used for determination of input beta, electric field and phase of linac modules. The program partially automates a procedure in which beam phase is measured versus module phase over wide ranges of module phase. A chi-squared minimization is performed to obtain the best curve fit between the measured points and a function which calculates single-particle transport through the module. The adjustable parameters for the curve fit are input beta, electric field, and phase offset for the module being tuned. After curve fitting, the module phase is determined in relation to its position along the phase-scan curve.

Description

The phase scan method for linac tuning was described in reference 1. Basically, linac module phase is scanned over approximately 360 degrees while the phase of the rf signal induced by the beam in a strip-line detector is monitored. The beam detector is located at the end of a free-drift region downstream of the module being examined. As the module phase varies, the acceleration of the beam through the module changes in a characteristic manner, which can be calculated. The changes in acceleration result in changes in the time of flight of the beam as it passes the beam monitor. Changes in the time of flight are proportional to phase changes in the induced rf signals at the beam monitor. Module electric field and input beta may be determined by matching the experimental curve of beam phase versus module phase with a theoretical curve.

The program for accomplishing this task is presently located on parameter page L16 of the console control system. A listing of the program, containing numerous comments, is given at the end of this report. Figure 1 shows the user interface for the program. The box on the left contains input parameters that the operator must input before each run. The box in the upper right contains output parameters that may be used to diagnose the progress of the program as it runs. The "ACTION" box

is a pull-down menu of possible operator actions.

The shaded area near the lower left of the interface prints the value of the electric field, and input beta offset from design. These are the principal parameters that are calculated in the program. The phase offset can have various meanings as the program runs. A message is printed after the phase-offset value indicating its present meaning. Finally, the bracketed message area at the bottom of the interface contains popup messages of various kinds. Operator instructions are given in this area and are always capitalized. Status reports are given in lower-case letters in the message area. Over the whole interface panel, green colors indicate all is well and the program may proceed. A yellow color indicates an unverified change has been made to a parameter in the input parameter list. If a yellow message appears in the message area, something unusual has occurred. In all cases, yellow indicates you may proceed, but with caution. A red color usually indicates that something needs attention before continuing the program. Purple is used for start-up or default parameters that have not been verified.

The first four parameters in the input parameter box on the left are the sines and cosines of the beam and tank (or module) phase. These readings originate in the I/Q demodulators near the linac. Occasionally, the traces produced by the program may appear inverted. This can be corrected by interchanging sine and cosine signals. The current values for these parameters are given in the output parameter box on the right of the interface panel.

"Gate Device", shown in the input box, is any device that gives an indication of the presence of the beam in the linac. The current reading for the gate signal is given in the output parameter box. If the program appears to have stopped, check this parameter first. The program will take data only when it senses the presence of the beam. A gate level which triggers data-taking may be set in the input parameter box.

"Tank Number" in the input parameter box must be in the range 03-17. The leading zero indicates an Alvarez tank specified by the second digit. A leading 1 indicates a side-coupled module specified by the second digit. The leading 1 or 0 must be included in the tank designation.

The "Monitor Number" indicates the strip-line detector or wall gap monitor which is used to detect beam phase. Currently, this monitor must either follow the module being tuned, or be located one module downstream of the module being tuned. Eventually, this parameter will include monitor locations more than one module downstream of the module being tuned. The longer free-drift regions will permit greater measurement sensitivity. The monitor number designates the module that the monitor follows.

"Setting Device" is the name of the device which changes the phase of the module being tuned. "Start Setting" is the initial module phase setting and stop phase is the final module phase setting. The phase range should normally be 360 degrees, or as close as possible to 360 degrees. The Alvarez tanks currently are not adjustable over a full 360 degrees so a smaller range will have to suffice for now. The scan increment determines the step size for the module phase setting. A value of 0.1 produces very high phase-scan resolution. "Sample Rate" is the value in Hertz.

In the output parameter box, "Sync Value" indicates whether or not all data was taken at the same instant in time. A value of 1 indicates data was taken at the same time, while zero indicates data was not taken at the same time. Data is accepted for plotting only if sync value equals 1. "No. Points", shown in the output box, is the number of points that are currently accepted for plotting. This parameter is used to diagnose progress of the program. If, for example, the current phase reading is not changing, but the number of points is steadily increasing, there may be a problem. It may also

indicate that the program is averaging over several fixed phase points, which it sometimes does when finding the average zero-power beam-phase reading. This latter event is normal.

Under the "ACTION" pull-down menu, "Accept Input" is selected when parameters in the input box are accepted by the operator. Some basic checks on the parameter values are made to be sure values are within acceptable ranges. Checks are made of device inputs to be sure device indices are assigned, indicative of valid device specifications. If all parameters pass this basic screening, then parameter colors change to green, if they are not already green.

Instructions next appear in the message area, directing the operator to turn off the proper sequence of tanks. Once this has been done, the program checks that the tanks have been turned off and then collects about 100 readings of beam phase. Zero-power data are plotted so that the operator can monitor results and look for any unusual behavior, such as noisy data. Average beam phase for the unpowered tank is then calculated along with the standard deviation in the data. Results are written in the output area following the words "Zero" and "Sigma".

Next, instructions in the message area tell the operator to turn on the module being tuned. The program checks that this action was performed and then begins plotting the raw data for beam phase versus module phase for the powered tank.. Tank phase for this plot is the reading taken from the device used for adjusting the phase. Beam phase readings cover a range of only ± 180 degrees, because of the nature of the I/Q demodulator system. This is the reason for the jumps in the raw data.

After raw data has been gathered, "Curve Match" is normally selected from the "ACTION" pull-down menu. This selection removes the 360 degree phase jumps in the data and replots the data.. For this step in the program, tank phase readings are taken from I/Q demodulators connected to tank pickup loops. I/Q demodulator readings replace phase adjustment device readings. The replacement occurs because readings from the tank phase adjustment for the Alvarez linac are inaccurate over some of the phase range (see reference 1). The graphics cursor is moved into the graphics window, at this point. The experimental points may be moved around the screen in order to partially align the measured points with a theoretical target curve plotted on the graphics screen. A collection of plots of target curves for the Linac Upgrade is given in reference 2. The experimental points are moved continuously along the module phase axis by moving the graphics cursor horizontally. Along the beam-phase axis, points are moved in discrete jumps of 360 degrees by moving the cursor vertically.

If discontinuities in the data remain, select "Auto-Scale" after making the best manual fit to the data under the "Curve Match" selection. "Auto-Scale" will move each data point vertically by 360 increments to minimize the distance from each data point to the target curve. The assumption here is that the module electric field and input beta are close to design values. This procedure will not work if the module settings are far from design. At this point the measured points should look very much like the target curve, if the module settings are close to design. A final opportunity is given to manually curve-fit data to the target curve.

By selecting "Least Squares", a chi-squared minimization is performed to find the theoretical curve that most closely matches the measured points. The adjustable parameters in the theory are module electric field, input beta, and phase offset. A graphics screen appears in which several points are selected. These are the points through which the program will attempt to fit a theoretical curve. After successful chi-squared minimization, the electric field, input beta, and phase offset for the fitted curve are written in the shaded region at the lower left of the interface panel. The measured curve is then shifted by the calculated phase offset and the best fit theory curve is plotted on top of the measured points. An example of the graphics output for one experiment is shown in figure 2. In the

figure, the curve that appears shifted from the data points is the design target curve. In this case, the design curve deviates considerably from the actual data points. The best curve fit is the second continuous curve that lies on top of the data points in the figure. The best fit is found for an electric field 1.04 times design and an input beta 0.21% below design, as printed in the shaded area in the lower left of the interface panel in figure 1.

Finally, "Adjust Phase" may be selected to determine the offset from design of the current phase setting. After making this selection a graphics screen appears which contains a small red vertical line superimposed upon a graph of the phase-scan curve. The line indicates the current position of the module phase setting relative to the phase-scan curve. Numbers in the middle of the graph show the deviation in the current phase setting from the design value. The module phase may now be adjusted to zero on the curve if the electric field and input beta are close to design values. The module phase must be adjusted on a separate parameter page. No module phase adjustment can be made from within the program at this point. The program is finished after this selection, and the operator may exit the application.

References

1. T. L. Owens and M. B. Popovic, "Phase Scan Signature Matching for Linac Tuning", Fermilab Linac Upgrade Report LU-186, 18 Nov., 1992.
2. T. L. Owens, "Target Curves for Phase Scan Tuning of Linac Upgrade", Linac Memo TLO-33193, March 31, 1993.

Console Location 3,
PA:L16 PHASE SCAN TUNING

1-APR-1993 16:19

L16

```
Tank sine:1:ddCOS3
Tank cosine:1:ddSIN3
Beam sine:1:ddCOS2
Beam cosine:1:ddSIN2
=====
Gate Device:1:TO6IN
Gate Level:5
Tank Number:06
Monitor Number:7
Setting Device:1:ph6adj
Start Setting:2
Stop Setting:310
Scan Increment:0.20
Current Value:
```

```
Sample Rate:15
Set Tolerance:2
```

```
E Field offset: 1.041
Beta offset: .9979
Phase offset:-4.913
```

Offset of curv
MESSAGES

```
Tank Sine Reading:-1549
Tank Cos Reading: 2312
Beam Sine Reading: 1023
Beam Cos Reading: 154.1
Current Phase: 310.2
=====
No. Points: 909
Gate Reading: 9.326
Zero:-55.03 Sigma: 1.973
Sync. Value: 1
```

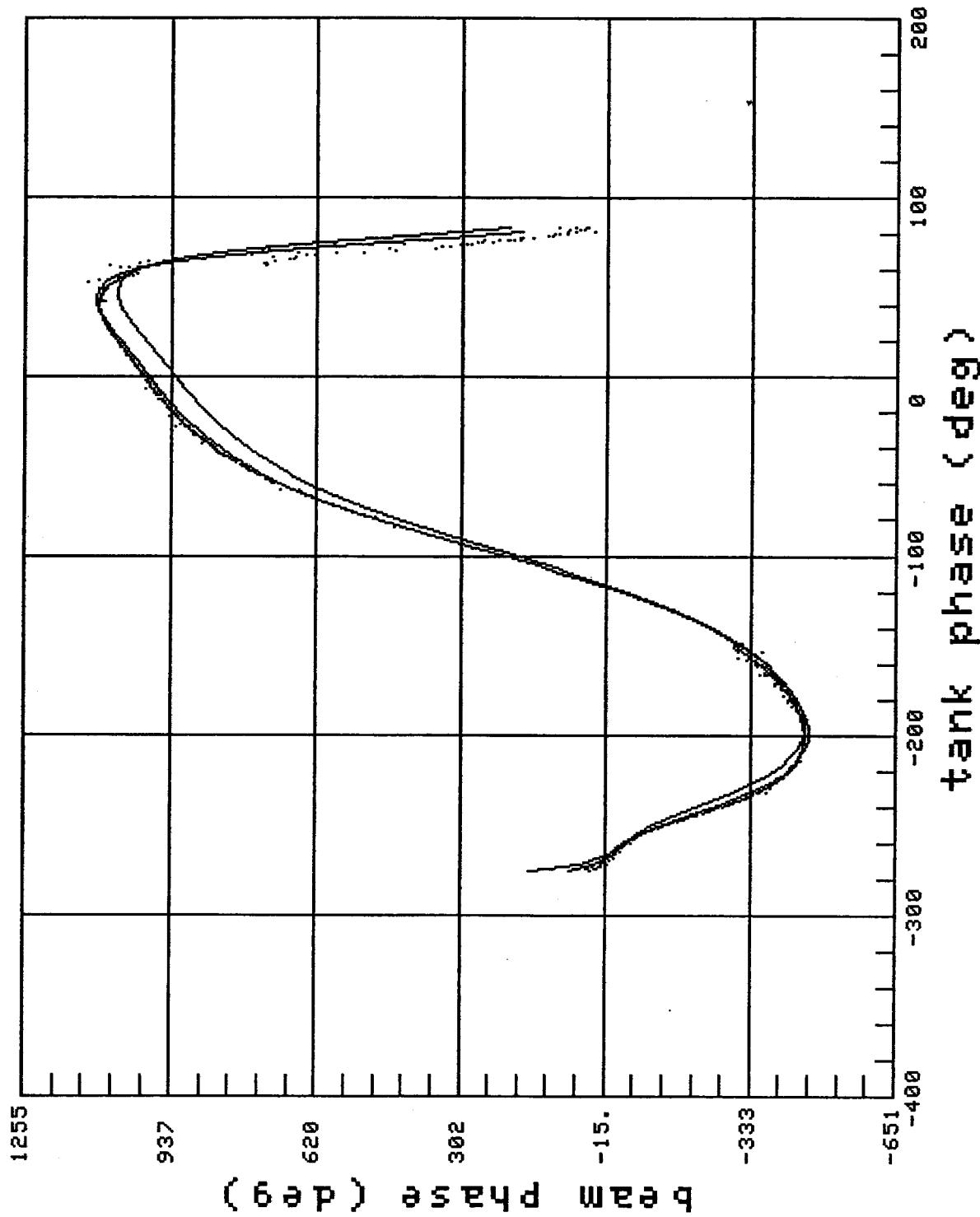
ACTION

```
SELECT ONE
Accept Input
Curve Match
Auto Scale
Save Data
Find <zero>
Least Squares
Adjust Phase
EXIT
```

Exiting module phase adjust

Console Location 3,
GxPA 1: None

1-APR-1993 16:16



C
C Copyright 1991, Universities Research Association. All rights reserved
PROGRAM PHASESCAN

C
C PROGRAM DESCRIPTION:
C

C
C PROGRAM FOR DETERMINING THE ELECTRIC FIELD AMPLITUDE AND PHASE,
C AND THE INPUT BETA IN EACH MODULE OF THE FERMILAB LINAC. MODULE PHASE
C IS SCANNED OVER ABOUT 360 DEGREES WHILE THE BEAM PHASE IS
C MEASURED AT A POINT DOWNSTREAM OF THE MODULE BEING TUNED.
C THE BEST THEORETICAL CURVE FIT TO THE EXPERIMENTAL CURVE IS
C MADE TO DETERMINE MODULE TUNE. THE ALVAREZ LINAC IS DESIGNATED
C BY MODULES 01-09, WHILE THE SIDE-COUPLED LINAC IS DESIGNATED
C BY MODULES 11-17. THE BEAM MONITORS ARE NUMBERED 1-9, DEPENDING
C ON WHICH MODULE THEY FOLLOW.

C
C
C AUTHOR:

C
C TOM OWENS, FERMILAB, PO BOX 500, BATAVIA, IL 60510
C (708) 840-4819

C CREATION DATE: JANUARY 2, 1993

C
C C H A N G E L O G

C Date | Name | Description

C -----+-----+
C 4/13/93 THIS VERSION IS COMPLETE INCLUDING UPGRADE MODULES
C AND ALVAREZ TANKS, BUT IT CONTAINS SIMULATION LINES AND
C IT USES IMSL ROUTINE DBCONF

10 INCLUDE 'clic\$include:cnsparam.inc' ! generic console defs

INTEGER*2 iwid, itype ! intype returned data
INTEGER*4 irow, icol, info
CALL WINDOW_INTTYPE(iwid,itype,irow,icol,info) ! decode possible interr
IF (itype .EQ. INTPER) THEN ! periodic
 CALL pgm_per
ELSE IF (itype .EQ. INTKBD) THEN ! keyboard
 CALL pgm_kbd
ELSE IF (itype .EQ. INTINI) THEN ! initialization
 CALL pgm_ini
ELSE IF (itype .EQ. INTTRM) THEN ! termination
 CALL pgm_trm
ENDIF
GOTO 10
END
!

```
C
C*****
C
C      SUBROUTINE pgm_ini
C+
C pgm_in
C
C Initialize program; called once after program is loaded.
C-
C
C      integer*4 stat2,iper,iscr,tanks,sflag,pflag,iflag
C      real tphis,tpi,rd,dr,pi,glim
C      double precision tha(5000),thc(5000)
C      real bzero,rdset,stngr,stmax,stmin
C      common/pplist/tphis,iper,tpi,rd,dr,tha,thc,iscr,glim
C      common/ini/tanks,bzero
C      common/after/rdset,stngr,stmax,stmin,sflag,pflag,iflag
C      tanks=0
C      iper=1
C      iscr=2
C      pi=4.0*atan(1.0)
C      tpi=2.0*pi
C      rd=180.0/pi
C      dr=1.0/rd
C
C      *** sflag=0 to search for zero
C
C      sflag=0
C
C      *** error routines.
C
C      stat2=error_init(28,1,60,log_none,,,1)
C
C      *** initialize tank value for tank repeat sensor in pgm_kbd.
C
C      RETURN
C      END
!
!
```

```

C
C*****
C
C      SUBROUTINE pgm_kbd
C+
C      pgm_kbd
C
C      Decode keyboard interrupts.
C-
C          external dbconf,lsqrs
INCLUDE 'CLIB$INCLUDE:CNSPARAM.INC'
INCLUDE 'CLIB$INCLUDE:DBPROPS.INC'
INCLUDE 'CLIB$INCLUDE:DIOLIB.INC'
INCLUDE 'CLIB$INCLUDE:CBSLIB.INC'
INCLUDE 'CLIB$INCLUDE:CLIB.INC'
INCLUDE 'CLIB$INCLUDE:CNS_DATA_STRUCTS.INC'
include 'clib$include:acnet_errors.inc'
integer*4 stat1,stat2,stat3,idlst,dindx(6),di,npts
integer*4 ii,i,iper,imenu,ibox,tankx,tanks,sflag
integer*4 ipers,yinc,jj,id,pflag,bpmn,bpmns,tankp
integer*4 limtp,imark,iflag,ipk,ilp
integer*4 na,itp,iparam(7),iperr,gindx
integer*2 pindx(6),errs(6),winid,type,srate,iwid
character*8 dev(6),dset,gtnk
character*2 tank
character*1 bpm,tchar
real value,sini,sfnl,fnpts,stng,smax,bphi(500),tphi(500)
real bpmin,bpmax,tpi,rd,dr,datas,bzero,stmp,tinc,tincs
real stngr,stmax,stmin,nomin,toler,nomdf
real dphs,bsgma
double precision dpabz,dpacz
double precision tha(5000),thc(5000),psoln
double precision ypick(20),xpick(20)
double precision xguess(3),xlb(3),xub(3),xscale(3)
double precision fscale,rparam(7),xsoln(3),func
double precision lsqrs
common/ini/tanks,bzero
common/after/rdset,stngr,stmax,stmin,sflag,pflag,iflag
common/mark/imark,nomin,nomdf
common/lsq/tank,bpmn,iperr,tankx,xpick,ypick,ipk
C
C*** plot parameters.
C
integer*2 wnd,tp
integer*4 rw,cl,in,j,iscr
integer*4 wnid,ticks(4),nchar(2),iflg,row,col,info
real viewport(4),limits(4),tphis
real xcs,glim
common/lst/idlst,errs,dindx,stng,finc,di,stol,smax
common/pplist/tphis,iper,tpi,rd,dr,tha,thc,iscr,glim
common/tar/wnid
C
C*** dialog window data
C
real stype(3),smin(3),smax1(3),sval(3)
character*8 promt(3),abuf1,abuf2,abuf3,abuf4,abuf5
character*8 abuf6,abuf7
data smin/-1000.0,-1000.0,0.0/
data smax1/1000.0,1000.0,1000.0/

```

```

data stype/cnv_float,cnv_float,cnv_float/
data promt/'initial:',' final:',' incr:/'

c
c*** no interupts if still plotting.
c
c      if(iper.le.iscr)then
c          stat2=error_display('Cannot interrupt until finished plotting'
c 1,err_none,0,3)
c          go to 1000
c      endif
c
c*** read devices and change modified data to yellow after interrupt.
c
        stat2=btvm(3,18,dev(1),-8)
        stat1=in_window_field(wmngr_background,3,18,9)
        if(stat1.eq.true)then
            stat1=btvm(3,18,dev(1),8,yellow)
        endif
        stat2=btvm(4,18,dev(2),-8)
        stat1=in_window_field(wmngr_background,4,18,9)
        if(stat1.eq.true)then
            stat1=btvm(4,18,dev(2),8,yellow)
        endif
        stat2=btvm(5,18,dev(3),-8)
        stat1=in_window_field(wmngr_background,5,18,9)
        if(stat1.eq.true)then
            stat1=btvm(5,18,dev(3),8,yellow)
        endif
        stat2=btvm(6,18,dev(4),-8)
        stat1=in_window_field(wmngr_background,6,18,9)
        if(stat1.eq.true)then
            stat1=btvm(6,18,dev(4),8,yellow)
        endif

c
c*** read gate device.
c
        stat2=btvm(9,18,dev(5),-8)
        stat1=in_window_field(wmngr_background,9,18,9)
        if(stat1.eq.true)then
            stat1=btvm(9,18,dev(5),8,yellow)
        endif

c
c*** read device to be set.
c
        stat2=btvm(13,18,dset,-8)
        stat1=in_window_field(wmngr_background,13,18,9)
        if(stat1.eq.true)then
            stat1=btvm(13,18,dset,8,yellow)
        endif

c
c*** read tank number (11-17 for upgrade, 01-09 for Alvarez).
c
        stat2=btvm(11,18,tank,-2)
        stat1=in_window_field(wmngr_background,11,18,3)
        if(stat1.eq.true)then
            stat1=btvm(11,18,tank,2,yellow)
        endif
        stat2=ascii_to_numeric(tankx,2,tank,cnv_long)
        if((tankx.gt.17).or.(tankx.lt.2))then
            stat1=btvm(11,18,tank,8,red)

```

```

        stat2=error_display('Invalid tank number (02-17).'
1,err_none,0,3)
        endif
C
C*** read beam monitor number (1-7 for upgrade, 1-9 for Alvarez).
C*** designates monitor after tank or module.
C
        stat2=btvm(12,18,bpm,-1)
        stat1=in_window_field(wmngr_background,12,18,2)
        if(stat1.eq.true)then
            stat1=btvm(12,18,bpm,1,yellow)
        endif
        stat2=ascii_to_numeric(bpmn,1,bpm,cnv_long)
        if(tankx.lt.9)then
            tankp=tankx+1
            if((bpmn.ne.tankx).and.(bpmn.ne.tankp))then
                stat1=btvm(12,18,bpm,1,red)
                stat2=error_display('Invalid beam monitor number
1 (= tank# or tank#+1)',err_none,0,3)
            endif
        endif
        if(tankx.eq.9)then
            if((bpmn.ne.9).and.(bpmn.ne.0))then
                stat1=btvm(12,18,bpm,1,red)
                stat2=error_display('Invalid beam monitor number
1 (= 9 or 0)',err_none,0,3)
            endif
        endif
        if(tankx.gt.10)then
            if((bpmn.ne.(tankx-10)).and.(bpmn.ne.(tankx-9)))then
                stat1=btvm(12,18,bpm,1,red)
                stat2=error_display('Invalid beam monitor number
1 (= module-10 or module-9)',err_none,0,3)
            endif
        endif
    endif
C
C*** read numeric values and check that I/O was performed.
C
C
C*** start setting (degrees).
C
        stat2=btvm(14,18,abuf7,-8)
        stat1=in_window_field(wmngr_background,14,18,9)
        if(stat1.eq.true)then
            stat1=btvm(14,18,abuf7,8,yellow)
        endif
        stat2=ascii_to_numeric(sval(1),8,abuf7,cnv_float)
        if((sval(1).gt.360.).or.(sval(1).lt.-100.))then
            stat1=btvm(14,18,abuf7,8,red)
            stat2=error_display('Start is out of range (0-360 Degrees).'
1,err_none,0,3)
        endif
C
C*** stop setting (degrees).
C
        stat2=btvm(15,18,abuf1,-8)
        stat1=in_window_field(wmngr_background,15,18,9)
        if(stat1.eq.true)then
            stat1=btvm(15,18,abuf1,8,yellow)
        endif

```

```

stat2=ascii_to_numeric(sval(2),8,abuf1,cnv_float)
if((sval(2).gt.360.).or.(sval(2).lt.0.))then
  stat1=btvm(15,18,abuf1,8,red)
  stat2=error_display('Stop is out of range (0-360 Degrees).'
1,err_none,0,3)
endif
c
c*** gate level.
c
  stat2=btvm(10,18,abuf6,-8)
  stat1=in_window_field(wmngr_background,10,18,9)
  if(stat1.eq.true)then
    stat1=btvm(10,18,abuf6,8,yellow)
  endif
  stat2=ascii_to_numeric(glim,8,abuf6,cnv_float)
  if((glim.gt.100.).or.(glim.lt.0.))then
    stat1=btvm(10,18,abuf6,8,red)
    stat2=error_display('Gate threshold is out of range (0-100).'
1,err_none,0,3)
  endif
c
c*** scan increment (degrees).
c
  stat2=btvm(16,18,abuf2,-8)
  stat1=in_window_field(wmngr_background,16,18,9)
  if(stat1.eq.true)then
    stat1=btvm(16,18,abuf2,8,yellow)
  endif
  stat2=ascii_to_numeric(sval(3),8,abuf2,cnv_float)
  if((sval(3).gt.100.0).or.(sval(3).lt.0.0))then
    stat1=btvm(16,18,abuf2,8,red)
    stat2=error_display('Scan increment out of range (0-100).'
1,err_none,0,3)
  endif
c
c*** sample rate, srate, is in hz.
c
  stat2=btvm(20,18,abuf3,-2)
  stat1=in_window_field(wmngr_background,20,18,3)
  if(stat1.eq.true)then
    stat1=btvm(20,18,abuf3,2,yellow)
  endif
  stat2=ascii_to_numeric(srate,2,abuf3,cnv_short)
  if((srate.gt.15).or.(srate.lt.1))then
    stat1=btvm(20,18,abuf3,8,red)
    stat2=error_display('Sample rate is out of range (1-15 Hz).'
1,err_none,0,3)
  endif
c
c*** setting tolerance, stol, is absolute tolerance, not relative.
c
  stat2=btvm(21,18,abuf4,-8)
  stat1=in_window_field(wmngr_background,21,18,9)
  if(stat1.eq.true)then
    stat1=btvm(21,18,abuf4,8,yellow)
  endif
  stat2=ascii_to_numeric(stol,8,abuf4,cnv_float)
  if((stol.gt.360.).or.(stol.lt.0.1))then
    stat1=btvm(21,18,abuf4,8,red)
    stat2=error_display('Tol is out of range (.1-360 Degrees).'

```

```

1,err_none,0,3)
    endif
c
c*** check popup menu interrupts
c
175      continue
        ibox=in_window_box(wmngr_background,14,3,35,28)
c
c*** fortran_true==1
c
        if(ibox.eq.fortran_true)then
            stat1=popup_menu(17,40,8,16,'Accept Input      Curve Match
1      Auto Scale      Save Data      Find <zero>
&      Least Squares   Adjust Phase   EXIT
&,'SELECT ONE',imenu)
            if(imenu.eq.0)go to 175
            goto(171,172,176,173,174,181,191,1002)imenu
        endif
        go to 1000
171      continue
c
c*** change some of the display values green if data is accepted.
c
        stat3=btvm(11,18,tank,2,green)
        stat3=btvm(12,18,bpm,1,green)
        stat3=btvm(10,18,abuf6,8,green)
        stat3=btvm(15,18,abuf1,8,green)
        stat3=btvm(16,18,abuf2,8,green)
        stat3=btvm(20,18,abuf3,2,green)
        stat3=btvm(21,18,abuf4,8,green)
        stat3=btvm(14,18,abuf7,8,green)
c
c*** setup list of devices to read.
c
        pindx(1)=prread
        pindx(2)=prread
        pindx(3)=prread
        pindx(4)=prread
        pindx(5)=prread
        pindx(6)=prread
c
c*** set dev(6)=dset to find device index for setting device.
c
        dev(6)=dset
        stat1=dio_device_index(dev,dindx,6)
        if(stat1.ne.0)then
            stat2=error_display(' invalid devices',err_acnet,stat1)
            go to 1000
        endif
c
c*** read the nominal phase setting for the tank.
c
        stat1=dio_alarm_limits(dindx(6),nomin,toler,limtp)
c
c*** if read devices received valid indices, turn input lettering green.
c
        stat2=btvm(3,18,dev(1),8,green)
        stat2=btvm(4,18,dev(2),8,green)
        stat2=btvm(5,18,dev(3),8,green)
        stat2=btvm(6,18,dev(4),8,green)

```

```

        stat2=btvm(9,18,dev(5),8,green)
c
c*** sample rate, srate, is in 60 hz ticks
c
        srate=60/srate
        stat1=dio_bld_get(idlst,6,dindx,pindx,errs,srate)
c
c*** device for setting.
c
        stat1=dio_device_index(dset,di)
        if(stat1.ne.0)then
            stat2=error_display(' problem getting index',err_acnet,stat1)
            go to 1000
        endif
c
c*** if setting device received valid index turn input lettering green.
c
        stat2=btvm(13,18,dset,8,green)
c
c*** define setting range and minimum number of points.
c
c
        stat1=wrdspr(2,1,'initial: final: # pts.:',8,smin,smax1,sval,3
c     1,' setting parameters')
        istrt=sval(1)
        istop=sval(2)
c
c*** if a new tank or monitor was selected,
c*** change sflag to zero to look for new zero-power monitor signal.
c
        if((tankx.ne.tanks).or.(bpmn.ne.bpmns))sflag=0
c
c*** if sflag=1 the zero has already been found - skip zero finder.
c
        if(sflag.eq.1)go to 179
c
c*** setup for reading gradients
c
        if(tankx.lt.10)then
            stat2=numeric_to_ascii(tankx,1,tchar,cnv_long)
            if(stat2.ne.0)then
                stat2=error_display(' problem in ascii conversion
&of tank #-abort',err_acnet,stat1)
                go to 1000
            endif
            gtnk='L:GR'//tchar//'MID'
        else
c
c*** add parameter designations for upgrade module gradients here.
c
            continue
            !
        endif
        stat2=dio_device_index(gtnk,gindx,1)
        if(gindx.eq.0)then
            stat2=error_display(' problem getting gradient index-abort'
&,err_acnet,stat1,5,red)
            go to 1002
        endif
        ilp=0
298    stat2=error_display('INTERRUPT ANYWHERE AFTER TURNING
& OFF TANKS PAST TANK #'
    )

```

```

    &,err_int,tankx-1,-1,red)
c
c*** check that tank was turned off.
c
stat2=dio_get_dev(gindx,prread,grad)
c
c*** grad=0.0 temporarily during offline testing
c
grad=0.0
if(grad.gt.0.01)then
  stat2=error_display('Tank has not yet been turned off
&- tank#',err_int,tankx-1,5,yellow)
  if(ilp.gt.3)then
    stat2=error_display('ABORTING-too many tries to
& zero gradient',err_none,0,0,red)
    go to 1002
  endif
  ilp=ilp+1
  go to 298
endif
stat2=error_display('Tanks turned off-INTERRUPT ANYWHERE
& TO CONTINUE'
1,err_none,0,-1,green)

c
c*** take current reading of setting device.
c
stat1=dio_get_dev(dindx(6),prread,rdset)
stngr=stng
c
c*** display action before plotting zero.
c
stat2=error_display('Zero-power beam phase is being
1 plotted -STANDBY',err_none,0,0)
finc=-sval(3)
stmin=sval(1)
stmax=sval(2)
go to 180
c
c*** calculate average beam phase using tank off readings.
c
174  datas=0.0
do 190 i=1,iper-1
190  datas=thc(i)+datas
bzero=datas/(iper-1)
datas=0.0
do 193 i=1,iper-1
193  datas=dsqrt((thc(i)-bzero)**2)+datas
bsgma=datas/(iper-1)
stat1>window_display_value(wmngr_background,10,39,bzero
1,cnv_float,6,green)
stat1>window_display_value(wmngr_background,10,53,bsgma
1,cnv_float,6,green)
c
c*** once zero is found turn tank back on.
c
ilp=0
179  stat2=error_display('INTERRUPT ANYWHERE AFTER
&TURNING ON TANK #'
1,err_int,tankx,-1,red)
c

```

```

c*** check that tank was turned on.
c
    stat2=dio_get_dev(gindx,prread,grad)
    if(grad.lt.0.01)then
        stat2=error_display('Tank has not been turned on
&- tank#',err_int,tankx-1,5,yellow)
        if(ilp.gt.3)then
            stat2=error_display('ABORTING-too many tries to
& increase gradient',err_none,0,0,red)
            go to 1002
        endif
        ilp=ilp+1
        go to 179
    endif
    stat2=error_display('Tank turned on-INTERRUPT ANYWHERE
& TO CONTINUE',err_none,0,-1,green)

c
c*** display action before plotting data.
c
    stat2=error_display('Powered-beam phase is being
1 plotted -STANDBY',err_none,0,0)
    finc=sval(3)
180    continue

c
c*** input target curve.
c
    call target (tank,bpm,tphis,tphi,bphi,ii)

c
c*** set up plotting for pgm_per subroutine.
c
    stat1=wn_color(6)
    stat1=wn_connect_points(false)

c
c*** after each interrupt, set periodic counter, iper, to 1
c*** iflag=0 means tank phase does not start at 0
c
    iper=1
    imark=0
    iflag=0
1000    tanks=tankx
    bpmns=bpmn
    go to 1001
172    continue
    if(iper.eq.1)then
        stat2=error_display('Run PA before selecting curve match'
1,err_none,0)
        go to 175
    endif

c
c*** BEGIN CURVE MATCHING.
c
c
c*** first take difference from zero.

c
c
c!!!!!! simulation value for bzero !!!!
c
    bzero=0.0
    do 128 i=1,iper-1
128    thc(i)=thc(i)-bzero

```

```

c
c*** then take out 2*pi phase jumps.
c
c      call massage(thc,iper)
c      call massage(tha,iper)
c
c*** move the curve around using the graphics cursor.
c
c      call match (tank,bpm,tha,thc,iper)
c      go to 1001
c
c
1002    continue
        stat2=error_display(' ABORTING - MUST RERUN'
        &,err_none,0,0,red)
        stat2=error_display(' ABORTING - MUST RERUN'
        &,err_none,0,0,red)
c
c*** on abort, plot stops and program must be re-run from beginning
c*** To stop di=0. sflag=0 to re-search zero.
c
c      pflag=1
c      sflag=0
c      di=0
c      go to 1001
176     continue
c
c*** Auto-scale curve.
c
c      call scale (tank,tphi,bphi,ii)
c      call match (tank,bpm,tha,thc,iper)
c      go to 1001
181     continue
        xguess(1)=1.0
        xguess(2)=1.0
        xguess(3)=0.0
        data xlbtv/0.7,0.7,-25.0/,xubt/1.2,1.2,25.0/
        data xscale/1.0,1.0,1.0/,fscale/1.0/
        na=3
        itp=0
        iparam(1)=0
c
c*** pick points for least squares fitting
c
c      call pick(xpick,ypick,ipk)
c      stat2=error_display('Calculating least squares fit - STANDBY'
c      1,err_none,0,0,green)
c
c*** Erase "offset from nom.".
c
c      stat1=btvm(25,26,'Offset from nom.',16,black)
c      go to 999
c
c*** skip call to dbconf temporarily.
c
c      call dbconf(lsqr,na,xguess,itp,xlb,xub,xscale,fscale
c      &,iparam,rparam,xsoln,func)
c
c*** temporary solutions.
c

```

```

c999      continue
c          xsoln(1)=0.9
c          xsoln(2)=1.0
c          xsoln(3)=10.0
1         stat1=window_display_value(wmngr_background,23,18,xsoln(1)
1,cnv_float,6,red)
1         stat1=window_display_value(wmngr_background,24,18,xsoln(2)
1,cnv_float,6,red)
1         stat1=window_display_value(wmngr_background,25,18,xsoln(3)
1,cnv_float,6,red)
c
c*** Print "Offset of curves".
c
stat1=btvm(25,26,'Offset of curves',16,red)
stat1=wn_color(yellow)
c
c*** shift plot according to least squares prescription.
c
stat2=error_display('Plotting best theory curve - STANDBY'
1,err_none,0,0,green)
call target (tank,bpm,tphis,tphi,bphi,ii)
stat1=wn_color(6)
stat1=wn_connect_points(false)
do 209 i=1,iper-1
tha(i)=tha(i)+xsoln(3)
if(thai.gt.(tphis+360.0))tha(i)=tha(i)-360.0
if(thai.lt.tphis)tha(i)=tha(i)+360.0
209 stat1=wn_point(thai,thc(i))
stat1=wn_color(red)
stat1=wn_connect_points(true)
stat1=wn_draw_mode(lx_replace)
psoln=tphis
psinc=3.6
c
c*** Draw the best fit theory curve.
c
do 267 i=1,100
call dphase (xsoln(1),xsoln(2),psoln,dpabz,dpacz)
if((bpnn.eq.tankx).or.(bpnn.eq.tankx-10))then
dphs=dpabz
else
dphs=dpacz
endif
stat1=wn_point(psoln,dphs)
267 psoln=psoln+psinc
stat2=error_display('Plot of best curve fit completed'
1,err_none,0,0,green)
go to 1001
191 continue
c
c*** Show the current position of the module phase on the current
c*** plot. Update as the phase is changed.
c
call phadj
173 continue
1001 RETURN
END
!

```

```

C
C*****SUBROUTINE pgm_per*****
C
C      SUBROUTINE pgm_per
C
C+ pgm_per
C
C  Process periodic interrupts: ~15 Hz.
C-
C
INCLUDE 'CLIB$INCLUDE:CNSPARAM.INC'
INCLUDE 'CLIB$INCLUDE:DBPROPS.INC'
INCLUDE 'CLIB$INCLUDE:DIOLIB.INC'
INCLUDE 'CLIB$INCLUDE:CBSLIB.INC'
INCLUDE 'CLIB$INCLUDE:CLIB.INC'
integer*4 idlst,dindx(6),stat1,di,stime,iper,iscr,sflag
integer*2 errs(6)
integer*4 pflag,ipmax,wnid,imark,iflag,iperr,ii
integer*4 ipk
character*2 tank
double precision efld,bti,dpabz,dpacz
double precision ypick(20),xpick(20)
double precision tha(5000),thc(5000)
real vals(6),stng,finc,stol,smax,stngr,stmax,stmin
real tphis,tpi,rd,dr,cstng,glim,angle
real con2,tphase(5000),nomdf,nomin,plast
common/lst/idlst,errs,dindx,stng,finc,di,stol,smax
common/pplist/tphis,iper,tpi,rd,dr,tha,thc,iscr,glim
common/after/rdset,stngr,stmax,stmin,sflag,pflag,iflag
common/tar/wnid
common/mark/imark,nomin,nomdf
common/lsq/tank,bpmn,iperr,tankx,xpick,ypick,ipk
common/phad/plast,const
C
C*** after an interrupt set pflag=0 so that plotting is armed.
C
C      if(iper.eq.1)pflag=0
C
C*** if pflag=0 plotting may occur if device indices are assigned.
C
C      if(pflag.eq.1)go to 1000
C
C*** ipmax = maximum number of tries to reach extremes.
C
C      ipmax=5000
C      if((dindx(1).ne.0).and.(di.ne.0))then
C
C*** get device readings.
C
C      stat1=dio_get_lst(idlst,vals,errs,,stime)
C
C*** simulation input (to be removed when testing completed)
C
C      if(sflag.eq.0)go to 215
bt1=1.0
efld=0.9
tha(iper)=vals(6)
angle=vals(6)*dr
call dphase (efld,bti,tha(iper),dpabz,dpacz)

```

```

thc(iper)=dpacz
vals(1)=cos(angle)
vals(2)=sin(angle)
vals(3)=dcos(dpacz*dr)
vals(4)=dsin(dpacz*dr)
215    continue
c
c* end simulation
c
c
c*** DISPLAY DEVICE READINGS.
c
c
c*** Phase detector readings.
c
        stat1=window_display_value(wmngr_background,3,53,vals(1)
1,cnv_float,6,green)
        stat1=window_display_value(wmngr_background,4,53,vals(2)
1,cnv_float,6,green)
        stat1=window_display_value(wmngr_background,5,53,vals(3)
1,cnv_float,6,green)
        stat1=window_display_value(wmngr_background,6,53,vals(4)
1,cnv_float,6,green)
c
c*** gate device reading.
c
        stat1=window_display_value(wmngr_background,9,53,vals(5)
1,cnv_float,6,green)
c
c*** Check if there is beam before continuing.
c*** [vals(5)=gate device reading, glim=gate threshold]
c
        if(abs(vals(5)).lt.glim)then
            stat1=window_display_value(wmngr_background,9,53,vals(5)
1,cnv_float,6,red)
            go to 1000
        endif
c
c*** sample time reading.
c
        stat1=window_display_value(wmngr_background,11,53,stime
1,cnv_long,6,green)
        if(stime.ne.1)then
            stat1=window_display_value(wmngr_background,11,53,stime
1,cnv_long,6,red)
            go to 1000
        endif
c
c*** setting device reading.
c
        stat1=window_display_value(wmngr_background,7,53,vals(6)
1,cnv_float,6,yellow)
c
c*** iper reading
c
        stat1=window_display_value(wmngr_background,8,53,iper
1,cnv_long,6,blue)
c
c*** ARCTAN OF TANK PHASE DETECTOR READINGS..
c

```

```

c
*** if both arguments of atan2 are zero, error results.
** Check for this condition and set these points to their
*** previous value and skip calls to atan2.
c
    if((abs(vals(2)).lt.1.0e-7).and.
1(abs(vals(1)).lt.1.0e-7))then
        if(iper.eq.1)then
            stat2=error_display('No signal on first read-QUITTING'
1,err_none,sflag,0,red)
            go to 1000
        endif
        tha(iper)=tha(iper-1)
        thc(iper)=thc(iper-1)
        go to 556
    endif
    tha(iper)=rd*atan2(vals(2),vals(1))

c
*** ATAN2 OF BEAM PHASE READINGS.
c
*** if both arguments of atan2 equal zero, an error results. Check
*** for this condition for 'thc'. If true set 'thc' equal to its previous
*** value and also set 'tha' to its previous value, so point is valid.
c
    if((abs(vals(4)).lt.1.0e-7).and.
1(abs(vals(3)).lt.1.0e-7))then
        if(iper.eq.1)then
            stat2=error_display('No signal on first read-QUITTING'
1,err_none,sflag,0,red)
            go to 1000
        endif
        thc(iper)=thc(iper-1)
        tha(iper)=tha(iper-1)
        go to 556
    endif
    thc(iper)=rd*atan2(vals(4),vals(3))
556    continue
c
*** first time after interrupts, find const needed to shift tank phase
*** to coincide with beginning of theory curve.
c
    if(iper.eq.1)const=(tha(1)-tphis)
    if(iper.eq.1)con2=(vals(6)-tphis)
c
*** shift all tank phases so beginning of experimental points coincides
*** with beginning of theory curve.
c
    tha(iper)=tha(iper)-const
    tphase(iper)=vals(6)
c
*** save array index of tank phase next to nominal and the difference
*** in phase between the two.
c
    if((tphase(iper).gt.nomin).and.(imark.eq.0))then
        if(sflag.eq.1)then
            imark=iper
            nomdf=tphase(iper)-nomin
        endif
    endif
    if(sflag.eq.1)tphase(iper)=vals(6)-con2

```

```

C
c*** If phase readings exceed extremes - plotting is complete.
C
    goto(209,210)sflag+1
    stat2=error_display('pgm_per>Problem with value of sflag'
1,err_none,sflag,0,red)
    go to 1000
209    continue
C
c*** If min. phase could not be reached, stop plot.
C
    if(iper.gt.100)then
        if(tphase(iper).gt.(tphase(iper-90)+10.0*finc))then
            sflag=1
            pflag=1
            stat2=error_display('WARNING-Cannot reach minimum
1 phase or already at zero',err_none,0,0,yellow)
            go to 1000
        endif
    endif
C
c*** if tank phase is less than min., stop plot. If the tank phase starts below
c*** minimum (iflag=1), then take points until above phase movement sensor logic
C
    if(vals(6).lt.stmin)then
        if(iper.eq.1)iflag=1
C
c*** if tank phase starts at 0, then keep sflag=0 and pflag=0.
C
        if(iflag.eq.1)go to 214
        stat2=error_display('Zero power plot completed'
1,err_none,0,0,green)
C
c*** when zero scan is complete set sflag to 1 and pflag=1
C
        sflag=1
        pflag=1
        go to 1000
    endif
    go to 211
210    continue
C
c*** If max phase could not be reached, stop plot.
C
    if(iper.gt.100)then
        if(tphase(iper).lt.(tphase(iper-90)+5.0*finc))then
            sflag=0
            pflag=1
            stat2=error_display('WARNING-Cannot reach maximum
1 phase',err_none,0,0,yellow)
            go to 1000
        endif
    endif
C
c*** If max phase was exceeded, stop plot.
C
    if(vals(6).gt.stmax)then
        stat2=error_display('Powered plot completed'
1,err_none,0,0,green)
        sflag=0

```

```

pflag=1
c
c*** save last module phase reading for use in phase adjust subroutine.
c
plast=tha(iper-1)
go to 1000
endif
211 continue
if(iper.gt.ipmax)then
  stat2=error_display('Excessive # tries to reach phase
1 extremes- QUITTING',err_none,iper,0,red)
  go to 1000
endif
c
c*** plot point using setting device readings on abscissa.
c
214 continue
stat1=wn_point(tphase(iper),thc(iper))
iper=iper+1
iperr=iper
c
c*** if tank is already at min (iflag=1) do not attempt to set phase.
c
if(iflag.eq.1)go to 1000
c
c*** For Alvarz tanks use relative phase setting. For SCS use absolute.
c
if(tankx.lt.10)stng=finc
stat1=dio_set_dev(di,stng)
endif
if(stat1.ne.0)then
  stat2=error_display(' problem setting device',err_acnet,stat1)
endif
1000 RETURN
END
!

```

```

C
C*****
C
C      SUBROUTINE pgm_trm

c+ pgm_trm
C
C  Program termination. Called once, as program exits.
C-
        RETURN
        END

C
C
C
C*****
C
C      SUBROUTINE MASSAGE (pdet,npts)

C
C
C*** subroutine to eliminate 2*pi phase jumps in data array, pdet.
C
        double precision beam(5000),pdet(5000)
        real tpi,rd,dr
        integer*4 i,fn,npts
        pi=4.0*atan(1.0)
        tpi=2.0*pi
        rd=180.0/pi
        dr=1.0/rd

C
C*** pdet is currently in degrees, beam assumes radians, so convert.
C
        do 22 i=1,npts-1
22      beam(i)=pdet(i)*dr
        fn=0
        do 2 i=4,npts-1
        bcomp=(beam(i-1)+beam(i-2)+beam(i-3))/3.0-fn*tpi
        if((bcomp-beam(i)).gt.4.0)fn=fn+1.0
        if((bcomp-beam(i)).lt.-4.0)fn=fn-1.0
        beam(i)=beam(i)+fn*tpi
2       continue

C
C*** convert back to degrees.

C
        do 21 i=1,npts-1
21      pdet(i)=beam(i)*rd
        return
        end

C
C
C
C*****
C
C      SUBROUTINE TARGET (tank,bpm,tphis,tphi,bphi,ii)

C
C
C*** routine to plot target curve

C
        INCLUDE 'CLIB$INCLUDE:CNSPARAM.INC'
        INCLUDE 'CLIB$INCLUDE:DBPROPS.INC'

```

```

INCLUDE 'CLIB$INCLUDE:DIOLIB.INC'
INCLUDE 'CLIB$INCLUDE:CBSLIB.INC'
INCLUDE 'CLIB$INCLUDE:CLIB.INC'
integer*4 stat1,stat2,i,ii
character*2 tank
character*1 bpm
real tphis,tphi(500),bphi(500)
real bpmin,bpmax
integer*4 wnid,ticks(4),nchar(2)
real viewport(4),limits(4)
common/tar/wnid
open(unit=11,file='usr$disk3:[owens.targets]tar'
1//tank//bpm//'.dat',status='old')
ii=1
bpmin=10000.0
bpmax=-10000.0
rewind 11
224 read(11,222,end=223)tphi(ii),bphi(ii)
222 format(2x,e12.6,2x,e12.6)
c
c*** find min and max of theory curve for plotting.
c
if(bphi(ii).lt.bpmin)bpmin=bphi(ii)
if(bphi(ii).gt.bpmax)bpmax=bphi(ii)
ii=ii+1
go to 224
223 continue
close(unit=11)
c
c*** store first abscissa point so experiment can be referenced to it.
c
tphis=tphi(1)
c
c*** plot data.
c
data viewport/0.1,0.1,0.9,0.9/
bpmin=bpmin-200.0
bpmax=bpmax+200.0
limits(1)=-400.0
limits(2)=bpmin
limits(3)=200.0
limits(4)=bpmax
data ticks/6,5,6,5/
data nchar/4,4/
c
c*** setup plot
c
stat1=wn_screen_init('test')
stat1=wn_set_background_size_c(%val(512),%val(640))
stat1=wn_color(7)
stat1=wn_grid(wnid,viewport,limits,ticks,'tank phase (deg)'
1,2,6,'beam phase (deg)',2,6,nchar,wn_make_grid)
stat1=wn_color(5)
stat1=wn_connect_points(true)
stat1=wn_draw_mode(lx_replace)
c
c*** plot target curve
c
do 124 i=1,ii-1
124 stat1=wn_point(tphi(i),bphi(i))

```

```

return
end

C
C
C
C
C***** ****
C
C      SUBROUTINE MATCH (tank,bpm,tha,thc,iper)
C
C
C*** routine to allow curve matching experiment to target curves.
C

INCLUDE 'CLIB$INCLUDE:CNSPARAM.INC'
INCLUDE 'CLIB$INCLUDE:DBPROPS.INC'
INCLUDE 'CLIB$INCLUDE:DIOLIB.INC'
INCLUDE 'CLIB$INCLUDE:CBSLIB.INC'
INCLUDE 'CLIB$INCLUDE:CLIB.INC'
INCLUDE 'CLIB$INCLUDE:CNS_DATA_STRUCTS.INC'
integer*4 stat1,stat2,wnid,iper,j,jj,id,yinc
integer*4 rw,cl,in,imark
integer*2 wnd,tp
double precision tha(5000),thc(5000)
real tphis,xc,yc,tinc,tincs
real xcs,con,tphi(500),bphi(500),nomin,nomdf
real pmark,bmark,buper,blwer,thast,thadf
character*2 tank
character*1 bpm
common/tar/wnid
common/mark/imark,nomin,nomdf
    stat2=error_display('Curve matching -cursor moved to graphics
1 window',err_none,0,0)
    call target (tank,bpm,tphis,tphi,bphi,ii)
    stat1=wn_cursor_init(wnid,0.,0.)
    stat1=wn_color(6)
    stat1=wn_connect_points(false)
    stat1=wn_draw_mode(lx_xor)
    thast=tha(iper-1)

C
C*** replot.
C
do 138 i=1,iper-1
138  stat1=wn_point(thc(i),tha(i))
j=0
xcs=0.0
stat1=wn_cursor_update(id,xc,yc)
yinc=yc/75.0
tinc=yinc*360
tincs=tinc
xcs=xc

C
C*** MOVE THE PLOT AROUND.
C
105   stat1=window_intype(wnd,tp,rw,cl,in)
        stat1=wn_cursor_update(id,xc,yc)
        yinc=yc/75.0
        tinc=yinc*360

C
C*** wait for a change in cursor position before acting.

```

```

c           if((xc.eq.xcs).and.(tinc.eq.tincs))go to 111
c
c*** first erase plot.
c
do 108 i=1,iper-1
108   stat1=wn_point(thc(i),tha(i))
c
c*** shift plot according to cursor position.
c
do 109 i=1,iper-1
tha(i)=tha(i)+xc-xcs
thc(i)=thc(i)+tinc-tincs
if(tha(i).gt.(tphis+360.0))tha(i)=tha(i)-360.0
if(tha(i).lt.tphis)tha(i)=tha(i)+360.0
109   stat1=wn_point(thc(i),tha(i))
xcs=xc
tincs=tinc
continue
111   if(tp.eq.intkbd_lx)then
      stat2=error_display('Curve matching completed'
1,err_none,0,0)
      if(imark.eq.0)go to 1000
      pmark=tha(imark)-nomdf
      bmark=thc(imark)
      buper=bmark+50.0
      blwer=bmark-50.0
      stat1=wn_color(4)
      stat1=wn_vector(pmark,blwer,pmark,buper)
      stat1>window_display_value(wmngr_background,25,18,pmark
1,cnv_float,6,yellow)
      stat1=btvm(25,26,'Offset from nom.',16,yellow)
      go to 1000
      endif
      j=j+1
      if(j.gt.1000)go to 1000
      go to 105
1000  thadf=tha(iper-1)-thast
c       stat1>window_display_value(wmngr_background,17,39
c       1,thadf,cnv_float,6,cyan)
       return
       end
c
c
c*****SUBROUTINE SCALE (tank,tphi,bphi,ii)
c
c
c*** shifts each point by increments of 360 degrees to account for
c*** possible missing 360 increments. Assumes that the data must be
c*** approach the target curve to within less than 180 degrees at
c*** each point.
c
integer*4 aindx,ntpi,ii,iper,iscr
character*2 tank
double precision tha(5000),thc(5000)
real tphi(500),bphi(500),pinc
real fntpi,slope,yval,rmdr
common/pplist/tphis,iper,tpi,rd,dr,tha,thc,iscr,glim

```

```

pinc=tphi(2)-tphi(1)
do 1, i=1,iper-1
aindx=(tha(i)-tphi(1))/pinc+1
if(aindx.gt.ii)then
  stat2=error_display('scale>WARNING-aindx gt than allowed
1-points may be missed',1,aindx,0,red)
  go to 1000
endif
slope=(bphi(aindx+1)-bphi(aindx))/pinc
dphi=tha(i)-tphi(aindx)
yval=bphi(aindx)+slope*dphi
fntpi=(yval-thc(i))/360.0
ntpi=fntpi
rmdr=fntpi-ntpi
if(rmdr.gt.0.5)ntpi=ntpi+1
if(rmdr.lt.-0.5)ntpi=ntpi-1
thc(i)=thc(i)+ntpi*360.0
1
1000
return
end
C
C
C*****SUBROUTINE LSQRS (N,XS,FV)
C
C*** Least squares curve fit subroutine.
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
INTEGER*4 N,bpmn,tankx,stat1,iperr
dimension tha(5000),thc(5000),xs(3)
double precision ypick(20),xpick(20)
CHARACTER*2 TANK
real tphis,tpi,rd,dr,glim,fvsig
integer*4 iper,iscr,ipk
common/pplist/tphis,iper,tpi,rd,dr,tha,thc,iscr,glim
common/lsq/tank,bpmn,iperr,tankx,xpick,ypick,ipk
fv=0.0
C
C*** efld and bti are relative to design values (fractions of design)
C
efld=xs(1)
bti=xs(2)
delp=xs(3)
stat1>window_display_value(0,23,18,xs(1)
1,5,6,6)
stat1>window_display_value(0,24,18,xs(2)
1,5,6,6)
stat1>window_display_value(0,25,18,xs(3)
1,5,6,6)
do 100 i=1,ipk-1
phin=xpick(i)+delp
call dphase(epfl,bti,phin,dpabz,dpacz)
if((bpmn.eq.tankx).or.(bpmn.eq.tankx-10))then
  dphs=dpabz
else
  dphs=dpacz
endif
100
fv=fv+dsqrt((ypick(i)-dphs)**2)
fvsig=fv/(ipk-1)
stat1>window_display_value(0,25,53,fvsig

```

```

1,5,6,6)
    return
end

c
c
c
      subroutine dphase (efld,bti,phi,dpabz,dpacz)
      implicit double precision (a-h,o-z)
      parameter (nc=100)
      character*2 tank,idum
      integer*4 tankx,bpmn,iperr,ipk
      dimension tha(5000),thc(5000)
      dimension pcel(nc),bcel(nc),pcntr(nc)
      double precision xpick(20),ypick(20)
      common/lsq/tank,bpmn,iperr,tankx,xpick,ypick,ipk
      COMMON/PARAM/FREQ,EREST,PI,TPI,C,WAVL,TW,FNCEL
      if(tankx.lt.10)then
        OPEN(UNIT=4,FILE='USR$DISK3:[OWENS.DELTAT] '//
      &'dr.dat',access='direct',form='formatted'
      &,status='OLD',recl=43)
        read(4,34,rec=tankx-2) idum,d1,d2,dab
34      format(2x,a2,3(1x,d12.7))
        close(unit=4)
      else
        OPEN(UNIT=3,FILE='USR$DISK3:[OWENS.DELTAT] '//
      &'drs.dat',access='direct',form='formatted'
      &,status='OLD',recl=43)
        read(3,35,rec=tankx-10) idum,d1,d2,dab
35      format(2x,a2,3(1x,d12.6))
        close(unit=3)
      endif
      PI=4.0*datan(1.0d0)
      TPI=2.0*PI
      DR=PI/180.0
      RD=1.0/DR
      EREST=939.301
      C=2.99792458d8
      FREQ=2.012416d8
      if(tankx.gt.9) freq=4.0*freq
      WAVL=C/FREQ
      TW=TPI/WAVL
      EFNOM=2.56

c
C*** INPUT NOMINAL VALUES FOR FIELDS ,BETAS, PHASES AND NO.CELLS FOR TANKS.
c
      GO TO (1,2,3,4,5,6,7,8,9,10
      &,11,12,13,14,15,16,17)tankx
1      GO TO 150
2      GO TO 150
3      FNCEL=35.0
      EFNOM=2.6
      PHNOM=-2.120960D02
      BTNOM=0.274699D+00
      GO TO 150
4      FNCEL=29.0
      EFNOM=2.6
      PHNOM=-2.120200D02
      BTNOM=0.356974D+00
      GO TO 150
5      FNCEL=24.0

```

```

PHNOM=-2.119820D02
BTNOM=0.414122D+00
GO TO 150
6 FNCEL=22.0
PHNOM=-2.120460D02
BTNOM=0.456921D+00
GO TO 150
7 FNCEL=21.0
PHNOM=-2.119380D02
BTNOM=0.491280D+00
GO TO 150
8 FNCEL=20.0
PHNOM=-2.119480D02
BTNOM=0.520413D+00
GO TO 150
9 FNCEL=19.0
PHNOM=-2.119540D02
BTNOM=0.545225D+00
10 go to 150
11 phnom=-1.22024d02
btom=0.456922d00
efnom=8.07d0
GO TO 150
12 phnom=-1.22153d02
btom=0.509397d00
efnom=7.85d0
GO TO 150
13 phnom=-1.22118d02
btom=0.555344d00
efnom=7.66d0
GO TO 150
14 phnom=-1.21994d02
btom=0.595601d00
efnom=7.48d0
GO TO 150
15 phnom=-1.21835d02
btom=0.630904d00
efnom=7.34d0
GO TO 150
16 phnom=-1.22109d02
btom=0.662048d00
efnom=7.20d0
GO TO 150
17 phnom=-1.22040d02
btom=0.689456d00
efnom=7.09d0
150 CONTINUE
if(tankx.gt.9)then
fncl=16.0
LCEL=65
else
LCEL=FNCEL+1
endif
C
C*** VALUES ARE RELATIVE TO DESIGN.
C
bta=btom*BTNOM
phia=phi+PHNOM
phia=phia*dr
ef=efld*EFNOM

```

```

if(tankx.lt.10)then
  call xfer(tank,bta,phia,ef,bcel,pcel,pcntr)
else
  call xfers(tank,bta,phia,ef,bcel,pcel,pcntr)
  continue
endif
phib=pcel(lcel)
btb=bcel(lcel)
va=bta*c
vb=btb*c
tabof=(dab+d1)/va
tacof=(dab+d2)/va
if(tankx.lt.10)then
  tabon=fncel/freq+d1/vb+(phib-phia)/(freq*tpi)
else
  tabon=(4.0*fncel+9.0)/(2.0*freq)+d1/vb+(phib-phia)
&/(freq*tpi)
endif
tacon=tabon+(d2-d1)/vb
dtabd=tabof-tabon
dtacd=tacof-tacon

```

C
C*** Keep phase values in degrees at 201.25 MHz.
C

```

if(tankx.gt.10) freq=freq/4.0
dpabz=dtabd*freq*360.0
dpacz=dtacd*freq*360.0
return
end

```

C
C*****
C

SUBROUTINE XFERS(TANK,BSTRT,PSTRT,EF0,BCEL,PCEL,PCNTR)

C
C
C SUBROUTINE XFERS CALCULATES BETA AND PHASE AT THE INPUTS TO EACH CELL
C OF A LINAC MODULE (4 SECTIONS, JOINED BY 3 DRIFT SPACES).
C A (BETA-LAMDA)/2 STRUCTURE IS ASSUMED. THE PHASE AT THE CENTER OF EACH GAP
C IS ALSO CALCULATED. CELL LENGTHS FOR EACH CELL OF A SECTION ARE ASSUMED
C CONSTANT. NUMBER OF CELLS PER SECTION IS GIVEN BY PARAMETER FNCEL.

C
C ASSUMES THE SPACE BETWEEN TANKS IS (3*BETA*LAMDA)/2. FOR OTHER
C SPACINGS CHANGE 3*PI TERM SEVERAL LINES AFTER STATEMENT LABEL 5
C TO N*PI WHERE N IS THE NUMBER OF BETA*LAMDA/2 INTERVALS IN THE
C DRIFT SPACE.

C
C INPUTS ARE:

BSTRT	BETA INTO FIRST CELL.
PSTRT	PHASE INTO FIRST CELL (RADIANs). NOTE THAT THIS PHASE SHOULD BE ABOUT -122 DEGREES FOR FERMI BETA LAMDA/2 STRUCTURE.
EF0	MODULE ELECTRIC FIELD (MV/M).
TANK	MODULE IDENTIFIER ("11"--"17" FOR LINAC UPGRADE MODULES) MUST BE DECLARED CHAR TANK*2 IN CALLING PROGRAM. (NOTE:TANK 18 IS LANL TANK 5)

C
C OUTPUTS ARE:

BCEL	ARRAY CONTAINING BETA VALUES AT EACH CELL INPUT.
PCEL	ARRAY CONTAINING PHASE VALUES AT EACH CELL INPUT.
PCNTR	ARRAY CONTAINING PHASE AT CENTER OF EACH CELL.

C
C NOTE: IF THERE ARE N CELLS, THE OUTPUT BETA AND PHASE OF THE
C MODULE WILL BE CONTAINED IN BCEL(N+1) & PCEL(N+1).
C

C
C IF THERE ARE M CELLS PER SECTION, THE OUTPUT BETA OF THE
C SECTION WILL BE CONTAINED IN BCEL(M+1). PCEL(M+1)
C CONTAINS ONLY THE PHASE INPUT TO THE NEXT SECTION.
C

C * THIS SUBROUTINE IS A MODIFICATION OF THE EARLIER XFER SUBROUTINE
C WHICH WAS WRITTEN FOR ALVAREZ STRUCTURES HAVING BETA*LAMDA=CELL LNGTH.
C
C

C *** WRITTEN BY T. L. OWENS ***
C MARCH 5, 1991
C

C*****
C

IMPLICIT DOUBLE PRECISION (A-H,O-Z)
PARAMETER (NC=100)
CHARACTER TANK*2, TFILE*7
COMMON/PARAM/FREQ, EREST, PI, TPI, C, WAVL, TW, FNCEL
COMMON/GLN/CLN(4)
DIMENSION BCEL(NC), PCEL(NC), PCNTR(NC), T(4), S(4), TP(4), SP(4)
1, TC(4), SC(4), TPC(4), SPC(4), SEP(4), BETAG(4)
TFILE='T'//TANK//'.DAT'
OPEN(UNIT=11, FILE='usr\$disk3:[owens.deltat] '//TFILE
&, STATUS='OLD')
READ(11,1)(SEP(J), CLN(J), J=1, 4)
FORMAT(2(2X, F6.5))
CLOSE(UNIT=11)
OPEN(UNIT=11, FILE='usr\$disk3:[owens.deltat] //' 'CF.DAT'
&, STATUS='OLD')
READ(11,2)(TC(J), SC(J), TPC(J), SPC(J), J=1, 4)
FORMAT(4(2X, F8.5))
CLOSE(UNIT=11)
BINI=BSTRT
PINI=PSTRT
NCELLS=FNCEL
CPR=TW*EFO/EREST
DWSUM=0.0
DBETASUM=0.0
GAMAI=DSQRT(1.0/(1.0-BINI**2))
WINI=EREST*(GAMAI-1.0)
PIO=PINI+PI/2.0
BCEN=0.0
BAV=BINI
GAV=DSQRT(1.0/(1.0-BAV**2))
BCEL(1)=BINI
PCEL(1)=PINI

C
C*** CALCULATE TRANSIT TIME FACTORS AND GEOMETRIC BETAS.
C

DO 4 I=1, 4
BTG=2.0*CLN(I)/WAVL
BETAG(I)=BTG
BTS=BTG**2
BTC=BTG*BTS
T(I)=TC(1)+TC(2)*BTG+TC(3)*BTS+TC(4)*BTC
S(I)=SC(1)+SC(2)*BTG+SC(3)*BTS+SC(4)*BTC
TP(I)=TPC(1)+TPC(2)*BTG+TPC(3)*BTS+TPC(4)*BTC
SP(I)=SPC(1)+SPC(2)*BTG+SPC(3)*BTS+SPC(4)*BTC

```

4      CONTINUE
C
C*** MAIN LOOP THROUGH MODULE. SECTION INDEX IS J. CELL INDEX IS I.
C
DO 6 J=1,4
DO 5 I=1,NCELLS
GINI=DSQRT(1.0/(1.0-BINI**2))
WINI=EREST*(GINI-1.0)
FL1=CLN(J)/2.0
FL2=FL1
P1=TW/BINI*FL1
VG=EF0*CLN(J)
C
C*** LOOP TO DETERMINE CENTRAL BETA
C
JFLAG=0
7      FPR=PI*VG*BETAG(J)/((BAV*GAV)**3*EREST)
C
C*** LOOP TO SOLVE TRANSCENDENTAL EQUATION FOR CENTRAL PHASE
C
IFLG3=0
33     PI0S=PIO
T1=FPR*(TP(J)*DSIN(PI0S)+SP(J)*DCOS(PI0S))
PI0=PINI+P1-T1
IFLG3=IFLG3+1
IF(IFLG3.EQ.1)GO TO 33
IF(ABS(1.0-PI0/PI0S).LT.1.0E-10)GO TO 30
IF(IFLG3.EQ.20)GO TO 32
GO TO 33
32     stat1=error_display('xfers> central phase does not converge'
&,1,0,5)
      GO TO 1000
30     CONTINUE
C
C*** END LOOP
C
BCENS=BCEN
BCEN=BINI+VG/(2.0*BINI*GINI**3*EREST)
1*(T(J)*DCOS(PI0)+S(J)*DSIN(PI0))
BAV=(BINI+BCEN)/2.0
GAV=DSQRT(1.0/(1.0-BAV**2))
JFLAG=JFLAG+1
IF(JFLAG.EQ.1)GO TO 7
IF(ABS(1.0-BCEN/BCENS).LT.1.0E-10)GO TO 8
IF(JFLAG.GT.20)GO TO 9
GO TO 7
9      stat1=error_display('xfers> central beta does not converge'
&,1,0,5)
      GO TO 1000
C
C*** END LOOP
C
8      CONTINUE
PCNTR(I+(J-1)*NCELLS)=PIO
DWTRM=TPI*(BETAG(J)/BCEN-1.0)*TP(J)
DW=VG*DCOS(PI0)*(T(J)-DWTRM)
DWSUM=DWSUM+DW
BT1=1.0+(WINI+DW)/EREST
BINIS=BINI
BINI=DSQRT(1.0-1.0/BT1**2)

```

```

BBAV=(BINIS+BINI)/2.0
GGAV=DSQRT(1.0/(1.0-BBAV**2))
F2=TW/((BINI*GINI)**3*EREST)
T2=FL2*(P1/FL1-F2*DW)
T3=TPI*VG*BETAG(J)/((BBAV*GGAV)**3*EREST)*TP(J)*DSIN(PIO)
PHIE=PINI+P1+T2-T3
156 PINI=PHIE-PI
BCEL(I+1+(J-1)*NCELLS)=BINI
PCEL(I+1+(J-1)*NCELLS)=PINI
5 CONTINUE
IF(J.EQ.4)GO TO 1000
C
C*** 3*PI TERM IS FOR 3*BETA*LAMBDA/2 SPACING BETWEEN TANKS.
C
6 PINI=PINI+TPI*FREQ*SEP(J+1)/(BINI*C)-3.0*PI
PCEL(1+J*NCELLS)=PINI
BCEL(1+J*NCELLS)=BINI
C
C*** END OF MAIN LOOP
C
1000 RETURN
END
C
C
C*****SUBROUTINE XFER(tank,BSTRT,PSTRRT,EFO,BCEL,PCEL,PCNTR)
C
C New Version
C CALCULATES BETA AND PHASE AT THE INPUT TO EACH CELL OF
C A LINAC TANK ,AND THE PHASE AT THE GAP CENTERS.
C
C INPUTS ARE:
C     BSTRT    BETA INTO FIRST CELL
C     PSTRRT   PHASE INTO FIRST CELL (RADIANs)
C     EFO      TANK ELECTRIC FIELD (MEGAVOLTS/METER)
C
C OUTPUTS ARE:
C     BCEl     ARRAY CONTAINING BETA VALUES AT EACH CELL INPUT
C     PCel     ARRAY CONTAINING PHASE VALUES AT EACH CELL INPUT
C     PCNTR   ARRAY CONTAINING PHASE AT THE CENTER OF EACH CELL
C
C NOTE: IF THERE ARE N CELLS, THE OUTPUT BETA AND PHASE OF THE
C TANK WILL BE CONTAINED IN BCEl(N+1) & PCel(N+1), RESPECTIVELY.
C
C *** WRITTEN BY T.L. OWENS ***
C           AUGUST 8, 1990
C
C*****
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
PARAMETER (NC=100)
character*2 tank
integer*4 stat2
COMMON/GEOM/CLN(NC),DT(NC),GAP(NC),BETAG(NC),GAMAG(NC)
COMMON/TRNST/T(NC),S(NC),TP(NC),SP(NC)
COMMON/PARAM/FREQ,EREST,PI,TPI,C,WAVL,TW,FNCEL
DIMENSION BCEl(NC),PCel(NC),PCNTR(NC)
call geo(tank)
BINI=BSTRT

```

```

PINI=PSTRT
NCELLS=FNCEL
CPR=TW*EF0/EREST
DWSUM=0.0
DBTASUM=0.0
GAMAI=DSQRT(1.0/(1.0-BINI**2))
WINI=EREST*(GAMAI-1.0)
PI0=PINI+PI
BCEN=0.0
BAV=BINI
GAV=DSQRT(1.0/(1.0-BAV**2))
BCEL(1)=BINI
PCEL(1)=PINI
C
C*** MAIN LOOP
C
DO 6 J=1,NCELLS
GINI=DSQRT(1.0/(1.0-BINI**2))
WINI=EREST*(GINI-1.0)
FL1=(DT(J)+GAP(J))/2.0
FL2=(DT(J+1)+GAP(J))/2.0
P1=TW/BINI*FL1
VG=EF0*CLN(J)
C
C*** LOOP TO DETERMINE CENTRAL BETA
C
JFLAG=0
7 FPR=PI*VG*BETAG(J)/((BAV*GAV)**3*EREST)
C
C*** LOOP TO SOLVE TRANSCENDENTAL EQUATION FOR CENTRAL PHASE
C
IFLG3=0
33 PI0S=PI0
T1=FPR*(TP(J)*DSIN(PI0S)+SP(J)*DCOS(PI0S))
PI0=PINI+P1-T1
IFLG3=IFLG3+1
IF(IFLG3.EQ.1)GO TO 33
IF(ABS(1.0-PI0/PI0S).LT.1.0E-10)GO TO 30
IF(IFLG3.EQ.20)GO TO 32
GO TO 33
32 continue
c     stat1=window_display_value(0,24,53,pio
c     1,5,6,5)
      stat2=error_display('xfer> central phase does not converge'
1,1,0,5)
      GO TO 1000
30 CONTINUE
C
C*** END LOOP
C
BCENS=BCEN
BCEN=BINI+VG/(2.0*BINI*GINI**3*EREST)
1*(T(J)*DCOS(PI0)+S(J)*DSIN(PI0))
BAV=(BINI+BCEN)/2.0
GAV=DSQRT(1.0/(1.0-BAV**2))
JFLAG=JFLAG+1
IF(JFLAG.EQ.1)GO TO 7
IF(ABS(1.0-BCEN/BCENS).LT.1.0E-10)GO TO 8
IF(JFLAG.GT.20)GO TO 9
GO TO 7

```

```

9      continue
      stat2=error_display('xfer>central beta does not converge'
1,1,0,3)
      GO TO 1000
C
C*** END LOOP
C
8      CONTINUE
PCNTR(J)=PIO
DWTRM=TPI*(BETAG(J)/BCEN-1.0)*TP(J)
DW=VG*DCOS(PIO)*(T(J)-DWTRM)
DWSUM=DWSUM+DW
BT1=1.0+(WINI+DW)/EREST
BINIS=BINI
BINI=DSQRT(1.0-1.0/BT1**2)
BBAV=(BINIS+BINI)/2.0
GGAV=DSQRT(1.0/(1.0-BBAV**2))
F2=TW/((BINI*GINI)**3*EREST)
T2=FL2*(P1/FL1-F2*DW)
T3=TPI*VG*BETAG(J)/((BBAV*GGAV)**3*EREST)*TP(J)*DSIN(PIO)
PHIE=PINI+P1+T2-T3
156      PINI=PHIE-TPI           ! BETA LAMBDA
BCEL(J+1)=BINI
PCEL(J+1)=PINI
6      CONTINUE
C
C*** END OF MAIN LOOP
C
1000    RETURN
END

```

```

C
C*****SUBROUTINE GEO(TANK)
C
C      GENERATES GEOMETRIC PARAMETERS FOR THE TANK INCLUDING
C
C      CLN      CELL LENGTH
C      DT       DRIFT TUBE LENGTH
C      GAP      CELL GAP
C      BETAG    GEOMETRIC BETA
C      GAMAG    GEOMETRIC GAMMA
C      T,S      FIRST ORDER TRANSIT TIME FACTORS
C      TP,SP    SECOND ORDER TRANSIT TIME FACTORS
C
C      TANK IS THE TANK NUMBER (01-12) - MUST BE DECLARED 'INTEGER*2'
C      IN CALL FROM MAIN PROGRAM.
C
C      CLN,DT,GAP ARE CONTAINED IN FILE 'TXX.DAT' XX=01-12
C      COEFFICIENTS FOR TRANSIT TIME FITS ARE CONTAINED IN 'CFXX.DAT'.
C
C          *** WRITTEN BY T.L. OWENS ***
C          AUGUST 8, 1990
C
C*****IMPLICIT DOUBLE PRECISION (A-H,O-Z)

```

```

PARAMETER (NC=100)
DIMENSION TC(4),SC(4),TPC(4),SPC(4)
COMMON/GEOM/CLN(NC),DT(NC),GAP(NC),BETAG(NC),GAMAG(NC)
COMMON/TRNST/T(NC),S(NC),TP(NC),SP(NC)
COMMON/PARAM/FREQ,EREST,PI,TPI,C,WAVL,TW,FNCEL/
CHARACTER TANK*2,TFILE*7,CFITS*11
TFILE='T'//TANK//'.DAT'
CFITS='CF'//TANK//'.DAT'
NCEL=FNCEL
OPEN(UNIT=4,FILE='USR$DISK3:[OWENS.DELTAT] '//TFILE
1,TYPE='OLD')
READ(4,1)(CLN(J),DT(J),GAP(J),J=1,NCEL+1)
FORMAT(3(2X,F6.5))
CLOSE(UNIT=4)
OPEN(UNIT=4,FILE='USR$DISK3:[OWENS.DELTAT] '//CFITS
1,TYPE='OLD')
READ(4,3)(TC(J),SC(J),TPC(J),SPC(J),J=1,4)
FORMAT(4(2X,F8.5))
CLOSE(UNIT=4)
C
C*** GENERATE TRANSIT TIME FACTORS, GEOMETRIC BETA, AND GEOMETRIC GAMMA
C
NCEL=FNCEL
DAB=0.0
DO 4 I=1,NCEL
DAB=DAB+CLN(I)
BTG=CLN(I)/WAVL
BETAG(I)=BTG
GAMAG(I)=DSQRT(1.0/(1.0-BTG**2))
BTS=BTG**2
BTC=BTG*BTS
T(I)=TC(1)+TC(2)*BTG+TC(3)*BTS+TC(4)*BTC
S(I)=SC(1)+SC(2)*BTG+SC(3)*BTS+SC(4)*BTC
TP(I)=TPC(1)+TPC(2)*BTG+TPC(3)*BTS+TPC(4)*BTC
SP(I)=SPC(1)+SPC(2)*BTG+SPC(3)*BTS+SPC(4)*BTC
4 CONTINUE
RETURN
END
C
C
C*****Subroutine phadj
C
C Subroutine to display the current phase setting on the current
C plot. Phase should be adjusted to zero for proper tuning.
C
INCLUDE 'CLIB$INCLUDE:CNSPARAM.INC'
INCLUDE 'CLIB$INCLUDE:DBPROPS.INC'
INCLUDE 'CLIB$INCLUDE:DIOLIB.INC'
INCLUDE 'CLIB$INCLUDE:CBSLIB.INC'
INCLUDE 'CLIB$INCLUDE:CLIB.INC'
INCLUDE 'CLIB$INCLUDE:CNS_DATA_STRUCTS.INC'
include 'clib$include:acnet_errors.inc'
double precision tha(5000),thc(5000)
real tphis,tpi,rd,dr,glim
real vals(6),stng,finc,stol,smax,phplt,phstr
integer*4 idlst,dindx(6),errs(6),di,stime
integer*4 iper,iscr,wnid,rw,cl,in,stat1,stat2
integer*4 ict

```

```

integer*2 wnd,tp
common/tar/wnid
common/pplist/tphis,iper,tpi,rd,dr,tha,thc,iscr,glim
common/lst/idlst,errs,dindx,stng,finc,di,stol,smax
common/phad/plast,const
stat2=error_display('ADJUST MODULE PHASE TO SET POINT'
&,err_none,0,0)
stat1=wn_cursor_init(wnid,0.,0.)
stat1=wn_color(4)
stat1=wn_draw_mode(lx_xor)
icount=0
phstr=0.0
105 stat1>window_intype(wnd,tp,rw,cl,in)
stat1=dio_get_lst(idlst,vals,errs,stime)

C
C*** ARCTAN OF TANK PHASE DETECTOR READINGS..
C
C
C*** if both arguments of atan2 are zero, error results.
C** Check for this condition and skip calls to atan2.
C
if((abs(vals(2)).lt.1.0e-7).and.
1(abs(vals(1)).lt.1.0e-7))then
    if(iper.eq.1)then
        stat2=error_display('No signal on first read-QUITTING'
1,err_none,vals(1),0,red)
        go to 1000
    endif
    stat2=error_display('phadj>Insufficient signal for atan'
1,err_none,vals(1),0,red)
    go to 556
    endif
    rphase=rd*atan2(vals(2),vals(1))-const
C
C*** simulation value for rphase -temporary.
C
rphase=vals(6)-const
delph=rphase-plast
cphs=tha(iper-1)+delph
if(icount.eq.0)cphss=cphs
if((cphs-cphss).gt.200.0)cphs=cphs-360.0
if((cphs-cphss).lt.-200.0)cphs=cphs+360.0
cphss=cphs
phplt=cphs
if(cphs.lt.tphis)phplt=cphs+360.0
if(cphs.gt.(tphis+360.0))phplt=cphs-360.0
C
C*** Skip replot if values are not changing.
C
C
stat1>window_display_value(wmngr_background,20,30,plast
1,cnv_float,8,yellow)
if(phplt.eq.phstr)go to 543
556 continue
if(icount.eq.0)go to 566
C
C*** First erase the old line, and numerical display.
C
stat1=wn_vector(phstr,0.0,phstr,100.0)
stat1=wn_display_value(0.0,150.0,phstr,cnv_float,8)

```

```

c
c*** Then plot the new line and display.
c
566      stat1=wn_vector(phplt,0.0,phplt,100.0)
      stat1=wn_display_value(0.0,150.0,phplt,cnv_float,8)
      phstr=phplt
543      continue
      if(tp.eq.intkbd_lx)then
         stat2=error_display('Exiting module phase adjust'
&,err_none,0,0)
         stat1=wn_cursor_disable()
         go to 1000
         endif
         icount=icount+1
         if(icount.gt.10000)then
            stat2=error_display('Too much time to adjust phase-QUITTING'
1,err_none,sflag,0,red)
            go to 1000
            endif
            go to 105
1000    return
            end
c
C*****
c
c          SUBROUTINE PICK (xpick,ypick,ipk)
c
c*** Subroutine for selecting points for least-squares fitting.
c
INCLUDE 'CLIB$INCLUDE:CNSPARAM.INC'
INCLUDE 'CLIB$INCLUDE:DBPROPS.INC'
INCLUDE 'CLIB$INCLUDE:DIOLIB.INC'
INCLUDE 'CLIB$INCLUDE:CBSLIB.INC'
INCLUDE 'CLIB$INCLUDE:CLIB.INC'
INCLUDE 'CLIB$INCLUDE:CNS_DATA_STRUCTS.INC'
include 'clib$include:acnet_errors.inc'
double precision tha(5000),thc(5000)
double precision xpick(20),ypick(20)
real tphis,tpi,rd,dr,glim,rw,cl
integer*4 iper,iscr,in,j,stat1,stat2,icnt,ipk,i
integer*2 wnd,tp
integer*4 py1,py2,px1,px2,ixscrn,iyscrn,jj,jk
real x1,x2,y1,y2
common/pplist/tphis,iper,tpi,rd,dr,tha,thc,iscr,glim
common/tar/wnid
      stat2=error_display('SELECT UP TO 20 POINTS FOR
& LEAST-SQUARES FIT (5 RECOMMENDED)'
1,err_none,green)
      stat1=wn_cursor_init(wnid,0.0,0.0)
      py1=450
      py2=425
      px1=71
      px2=170
c
c*** Important- "wn_vert" means that the coordinate is in the vertical
c*** plane (or y coordinate direction). Similarly, "wn_horz" means
c*** the x coordinate direction. True for "getworld" and "getscrn".
c
      stat1=wn_getworld(wn_vert,py1,y1)
      stat1=wn_getworld(wn_vert,py2,y2)

```

```

stat1=wn_getworld(wn_horz,px1,x1)
stat1=wn_getworld(wn_horz,px2,x2)
stat1=wn_fill_mode(lx_filled)
stat1=wn_box(x1,y1,x2,y2)
stat1=wn_color(yellow)
dx=x2-x1
dy=y1-y2
sx=dx*0.2
sy=dy*0.3
xt=x1+sx
yt=y2+sy
stat1=wn_text(xt,yt,'CONTINUE',9)
stat1=wn_color(red)
stat1=wn_set_plot_symbol(wn_triangle,4)
j=0
ipk=1
105 stat1=window_intype(wnd,tp,rw,cl,in)
if(tp.eq.intkbd_lx)then
  stat1=wn_cursor_update(id,xc,yc)
  if((xc.lt.x2).and.(xc.gt.x1)
&.and.(yc.gt.y2).and.(yc.lt.y1))then
    stat2=error_display('Point selection complete'
1,err_none,green)
    go to 1000
  endif
c
c*** Select data points that are closest to the current cursor position.
c
sep=abs(xc-tha(i))
if(sep.lt.sep)then
  icnt=i
  sep=sep
endif
1 continue
xpick(ipk)=tha(icnt)
ypick(ipk)=thc(icnt)
stat1=wn_symbol(thc(icnt),tha(icnt))

c
c*** If the same point was picked delete duplicate from collection.
c*** It won't appear on plot because of xor pixel logic.
c
do 546 jj=1,ipk-1
if(xpick(ipk).eq.xpick(jj))then
  ipk=ipk-1
  do 547 jk=jj,ipk
    xpick(jk)=xpick(jk+1)
    ypick(jk)=ypick(jk+1)
  endif
546 continue
  ipk=ipk+1
endif
if(j.gt.10000)then
  stat2=error_display('Pick> Too much time - EXIT'
1,err_none,0,red)
  go to 1000
endif
j=j+1
go to 105

```

```
1000    return  
end
```